Correlated Load Address Predictors

Michael Bekerman, Stephan Jourdan, Ronny Ronen,
Gilad Kirshenboim, Lihu Rappoport, Adi Yoaz, Uri Weiser

Intel Israel Architecture Research

ISCA-26 May 2, 1999

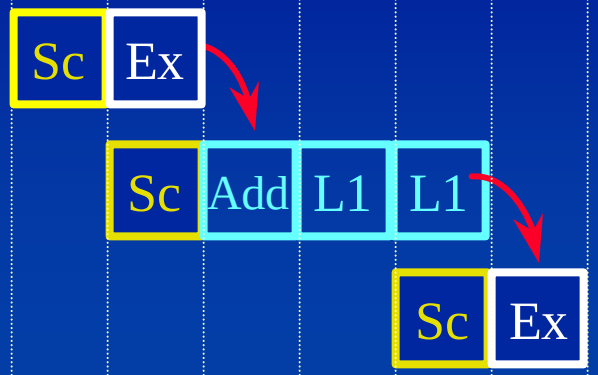
Agenda

- Motivation
- Load Behavior
- Context-based Address Predictor
- Hybrid Address Predictor
- Performance
- Pipeline Effects
- Conclusions

Motivation: Hide L1 Cache Hit Latency

Traditionally:

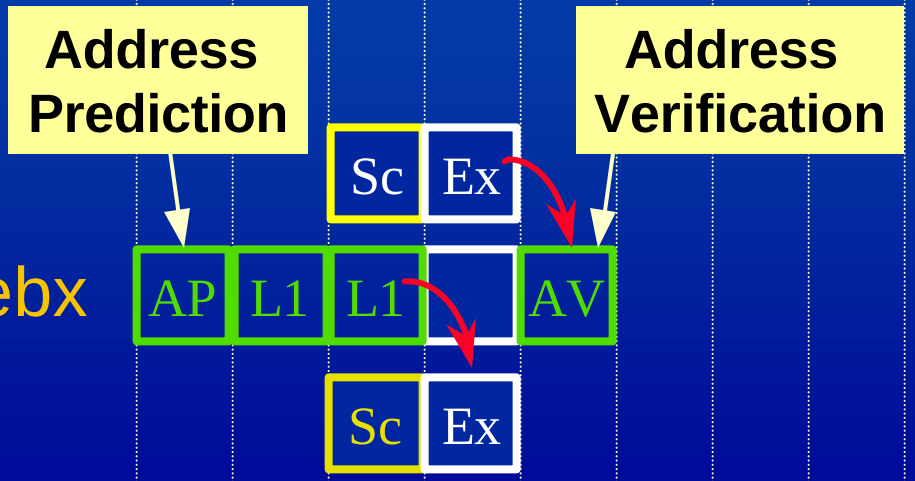
1) Def eax
2) Load (eax) → ebx
3) Use ebx



Sc - Schedule
Ex - Execute
Add - Address generate
L1 - L1 access
AP - Address Predict

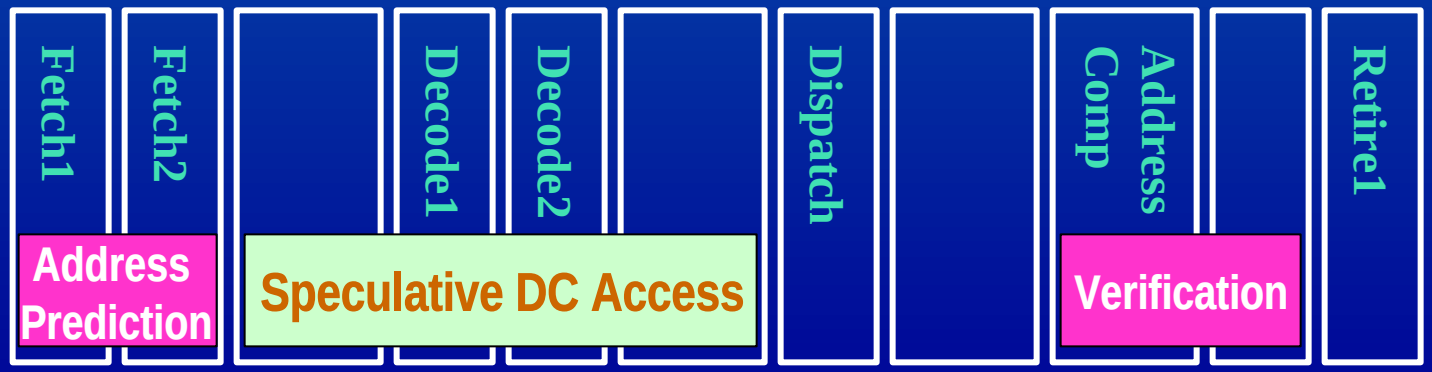
With Address Prediction:

1) Def eax
2) Load (eax) → ebx
3) Use ebx



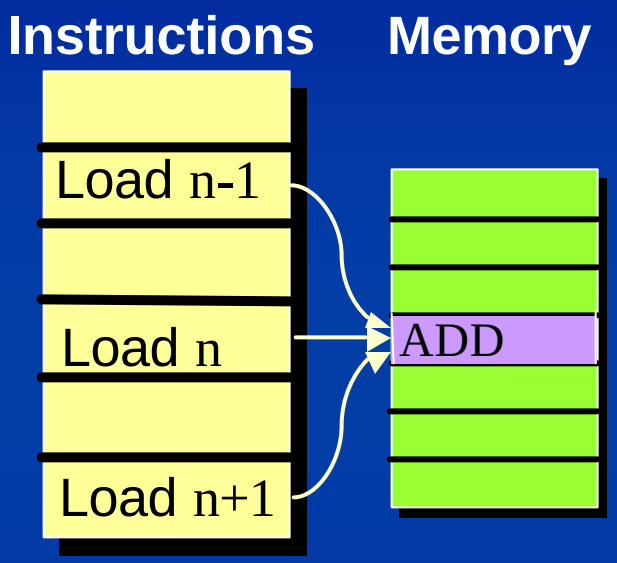
Where in the Pipeline?

- Prediction performed during the early pipe stages
- Speculative DCache access overlapped with front-end stages
  - Negative load-to-use delay
- Instructions dependent on predicted loads are speculatively scheduled
- Verification/recovery performed after address computation



Previous Address Prediction Schemes

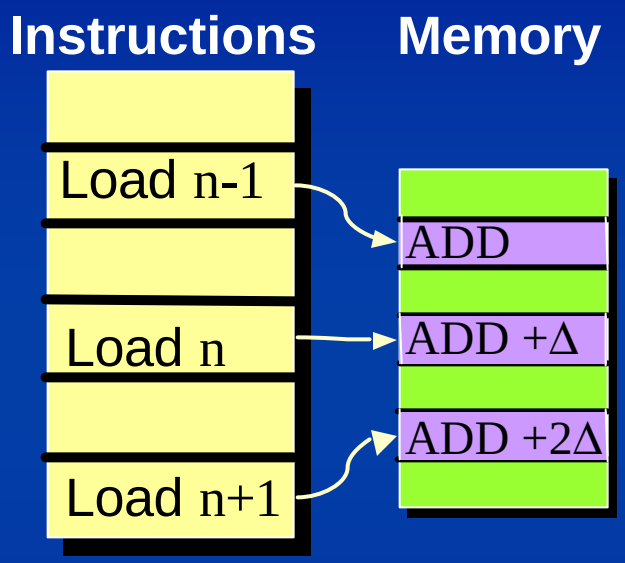
Constant Predictor



Predict $A_{n+1} = A_n$

40% of all loads

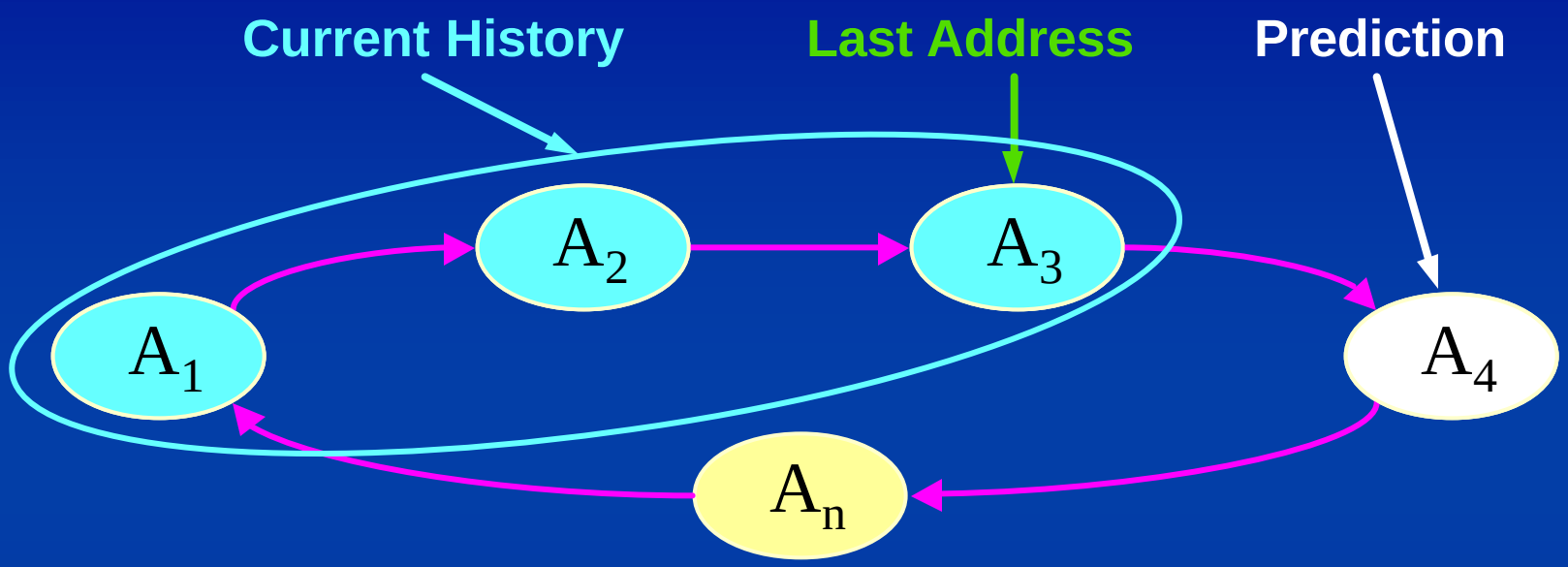
Stride Predictor



Predict $A_{n+1} = A_n + (A_n - A_{n-1})$

53% of all loads

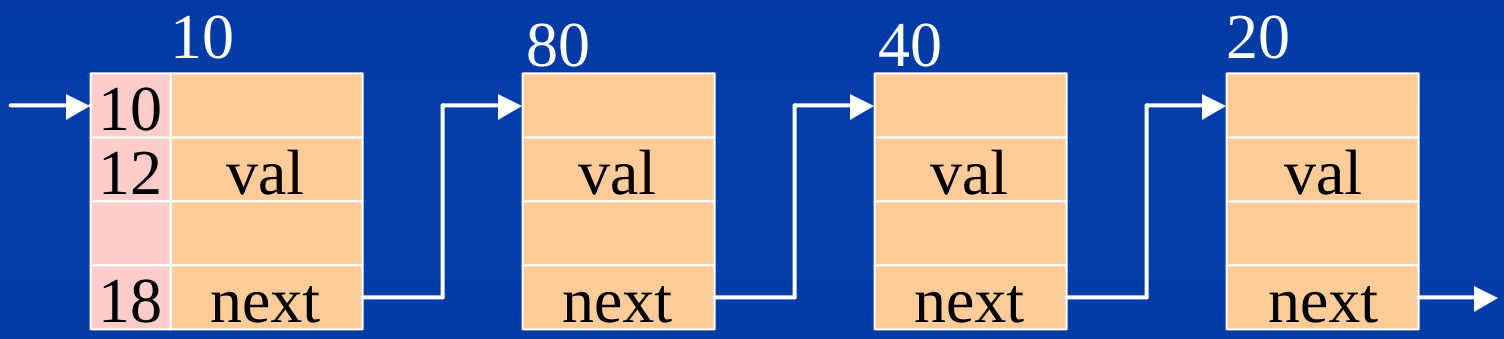
Context-Based Address Prediction



- Record load address sequences
- Predict the *next access* based on the current address history

Why Context-Based AP?

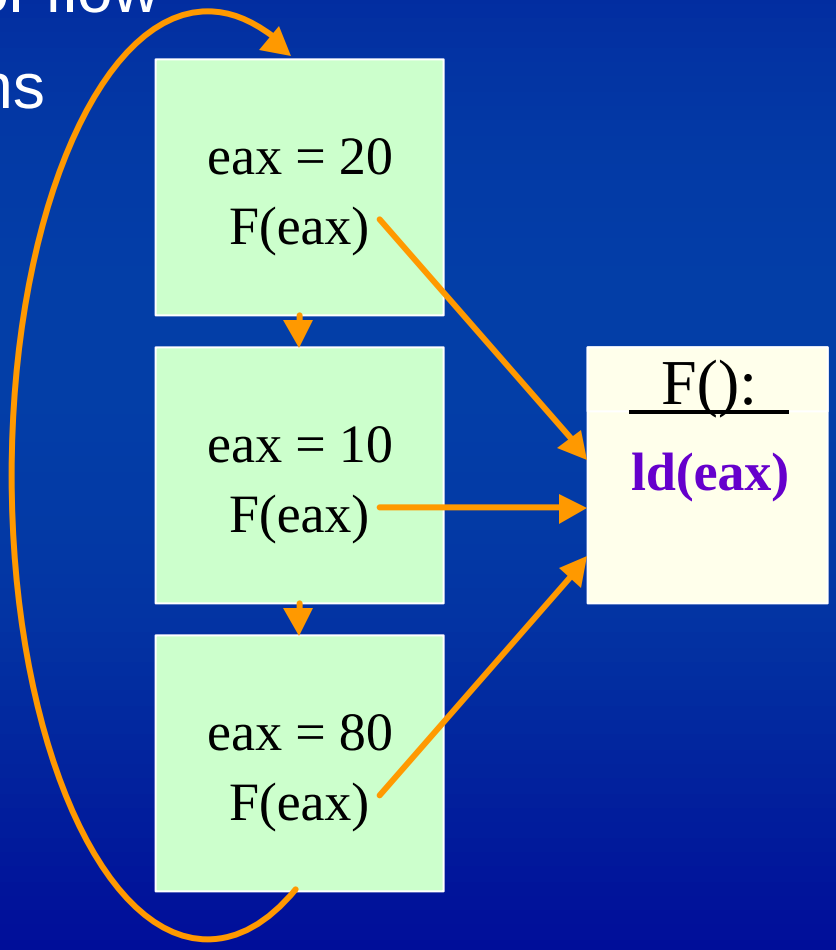
- Recursive Data Structures
  - Linked list
  - Binary tree



  - p = p->next
    - Load access pattern: 18, 88, 48, 28, ...
  - Context-Based AP mimics the load behavior

Why Context-Based AP?

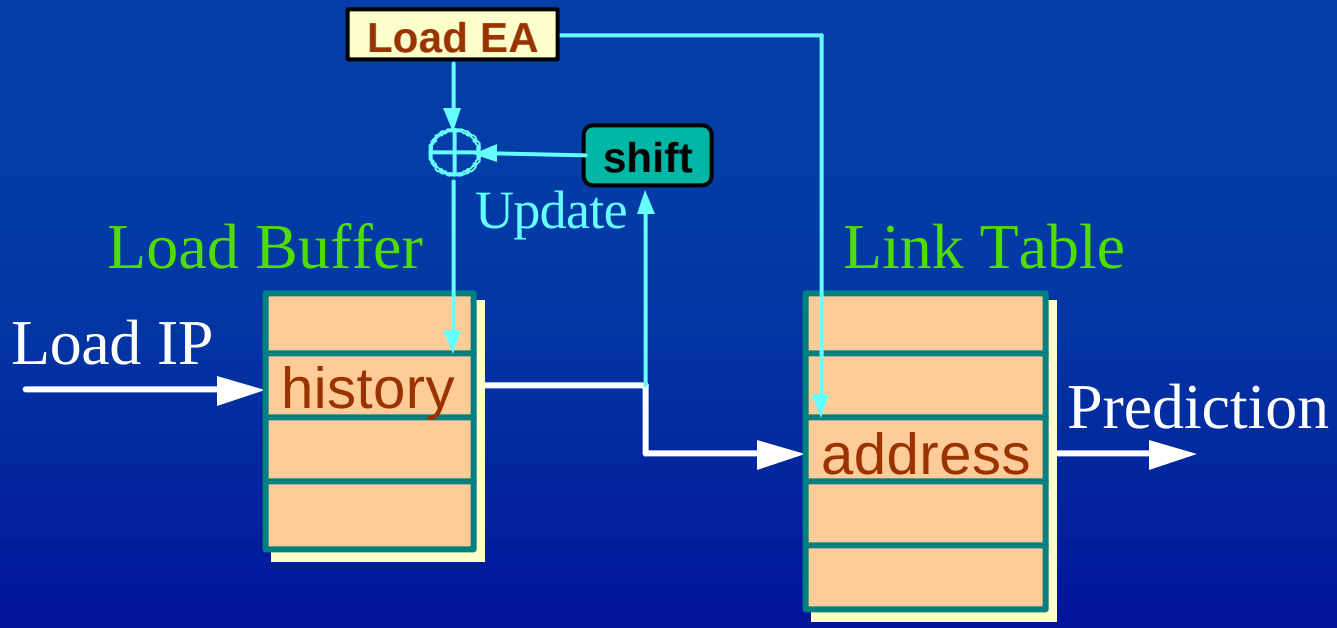
- Control Correlation
  - Repetitive and stable control-flow
  - Input parameters to functions
    - Values passed on the stack
    - Addresses passed through registers



History	Prediction
20,10,80	20
10,80,20	10
80,20,10	80

Context-Based Predictor Structure

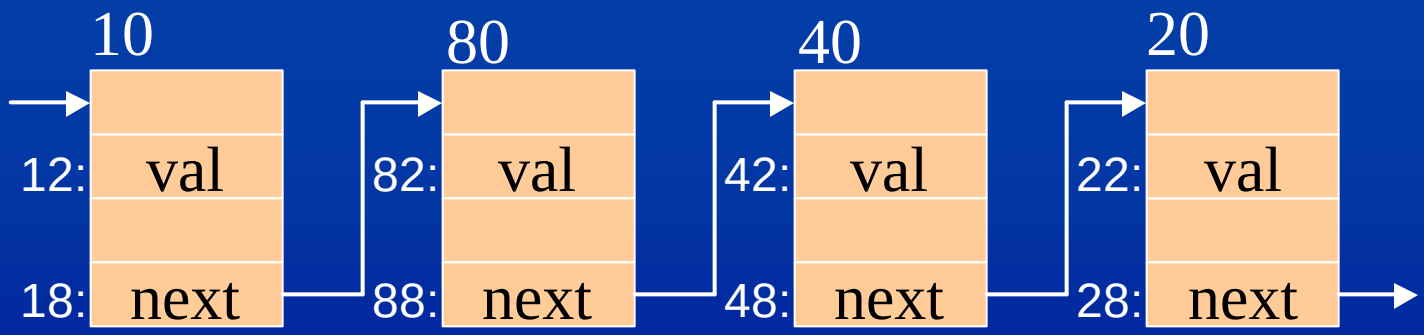
- Each *Load Buffer* entry records the history of recent addresses of the associated static load
  - Referenced by load IP
- The *Link Table* provides the next predicted address for current history



- Compacting addresses into a history vector
  - History length

Global Correlation

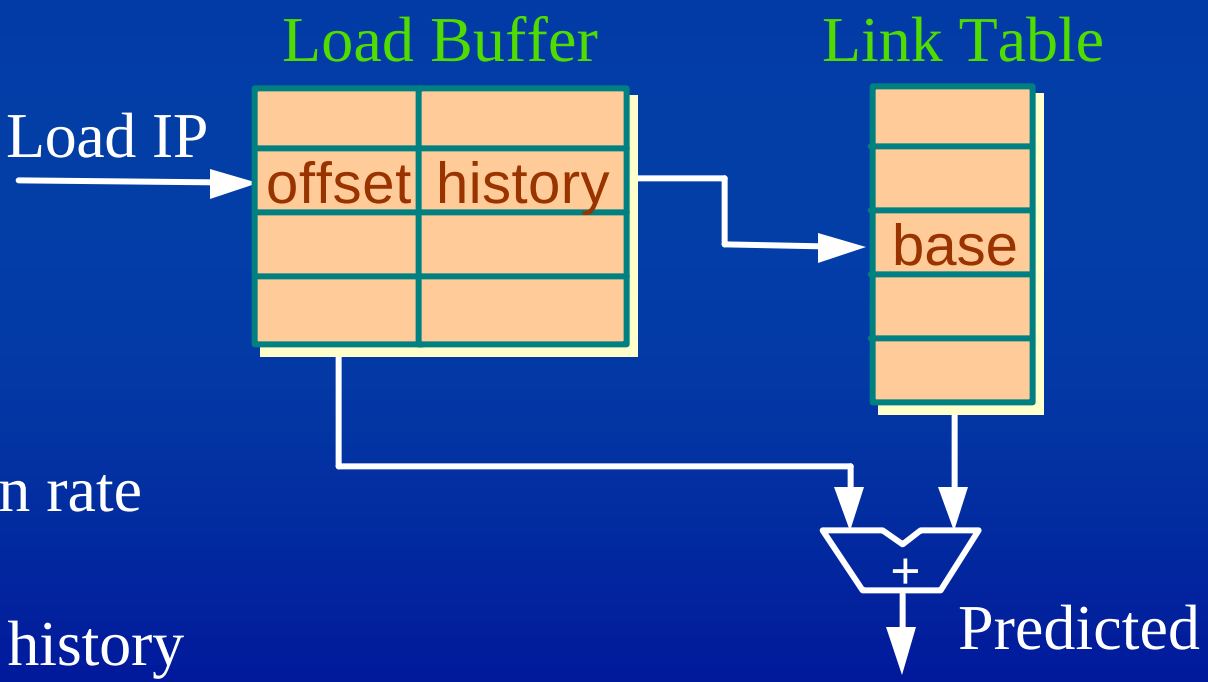
- Specific to Address Prediction
- **`val`** and **`next`** follow similar sequences:
  - structure base address: 10, 80, 40, 20, ...
  - **`val`** (offset 2): 12, 82, 42, 22, …
  - **`next`** (offset 8): 18, 88, 48, 28, …



- Represent load sequences accessing different fields of the same structure by the sequence of the base addresses and the offsets

Global Correlation

- Use base address instead of effective address
  - *base address = effective address - displacement*
  - **Predicted Address = *base address + offset***



Pros

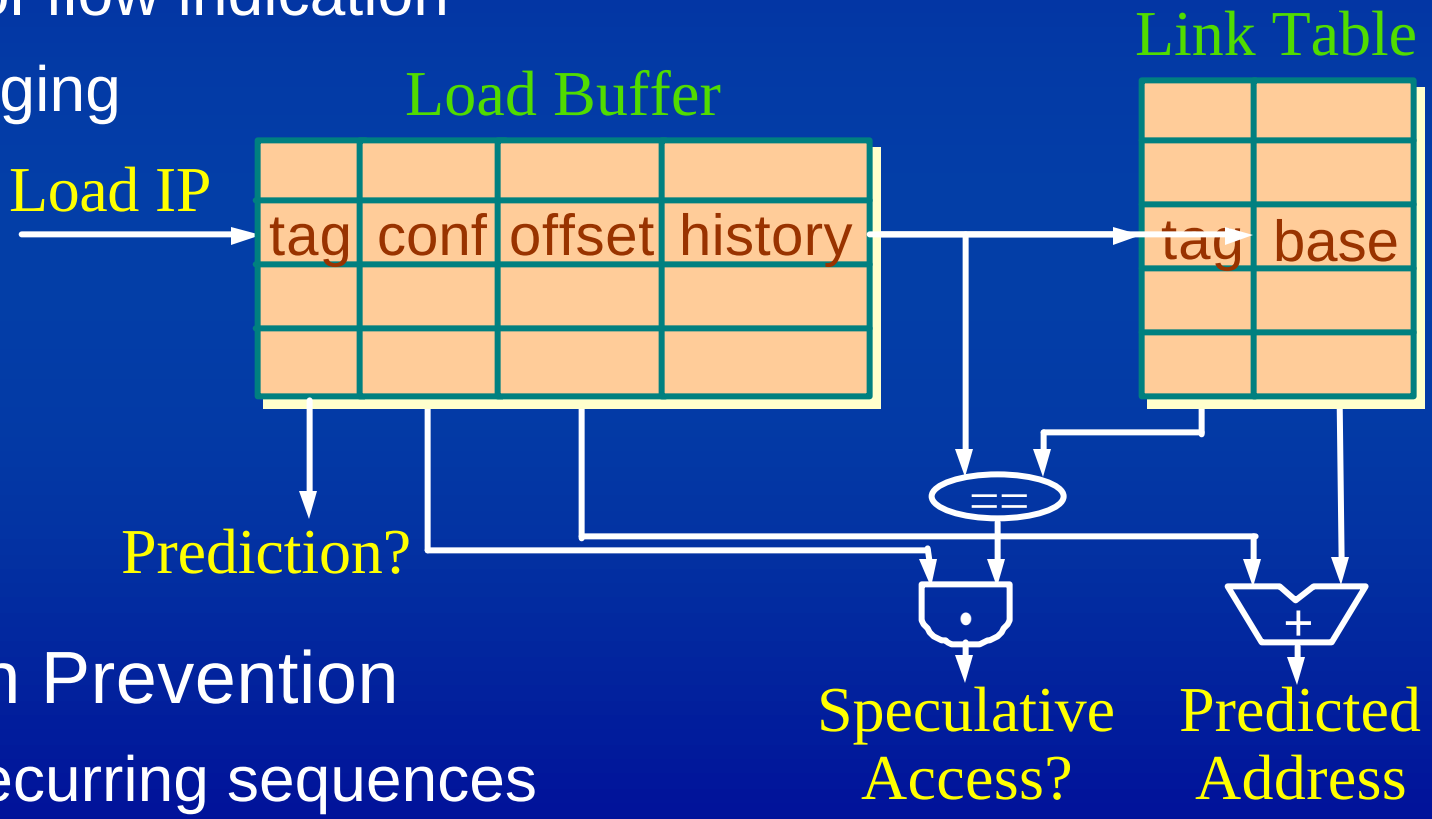
+ Requires less links
+ Improves prediction rate

Cons

- May require longer history
- Adds an adder to the prediction path

Enhanced Confidence Techniques

- Misprediction penalty is higher than no speculation
  - Saturating counters
  - Control-flow indication
  - LT tagging



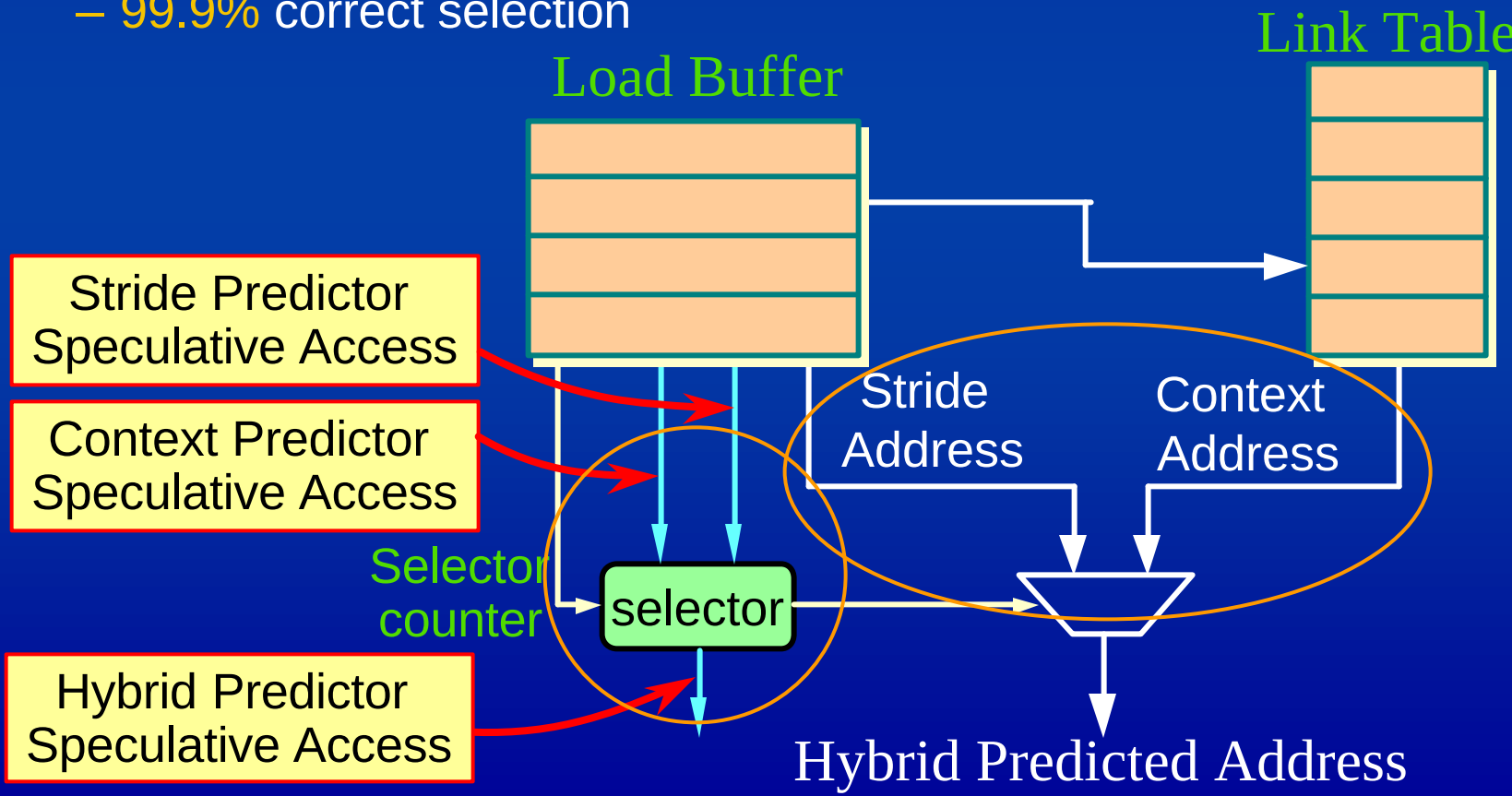
- Pollution Prevention
  - Non-recurring sequences
  - Long sequences

Hybrid Predictor

- Why a Hybrid address predictor?
  - Stride/constant predictor - only 50% of the loads
  - Context-based predictor - not cost-effective for simple loads
- Predict constant/stride loads with stride predictor (Load Buffer only)
- Predict sequences with context-based predictor (Load Buffer + Link Table)
- Share Load Buffer between stride and context-based predictors
  - New Load Buffer fields: *last address*, *stride*, state

Hybrid Predictor

- Predictor Selector
  - Choose only when both predictors are confident
  - Static selection: higher priority to one component
  - Dynamic selection: 2-bit saturating counter per LB entry
    - 99.9% correct selection

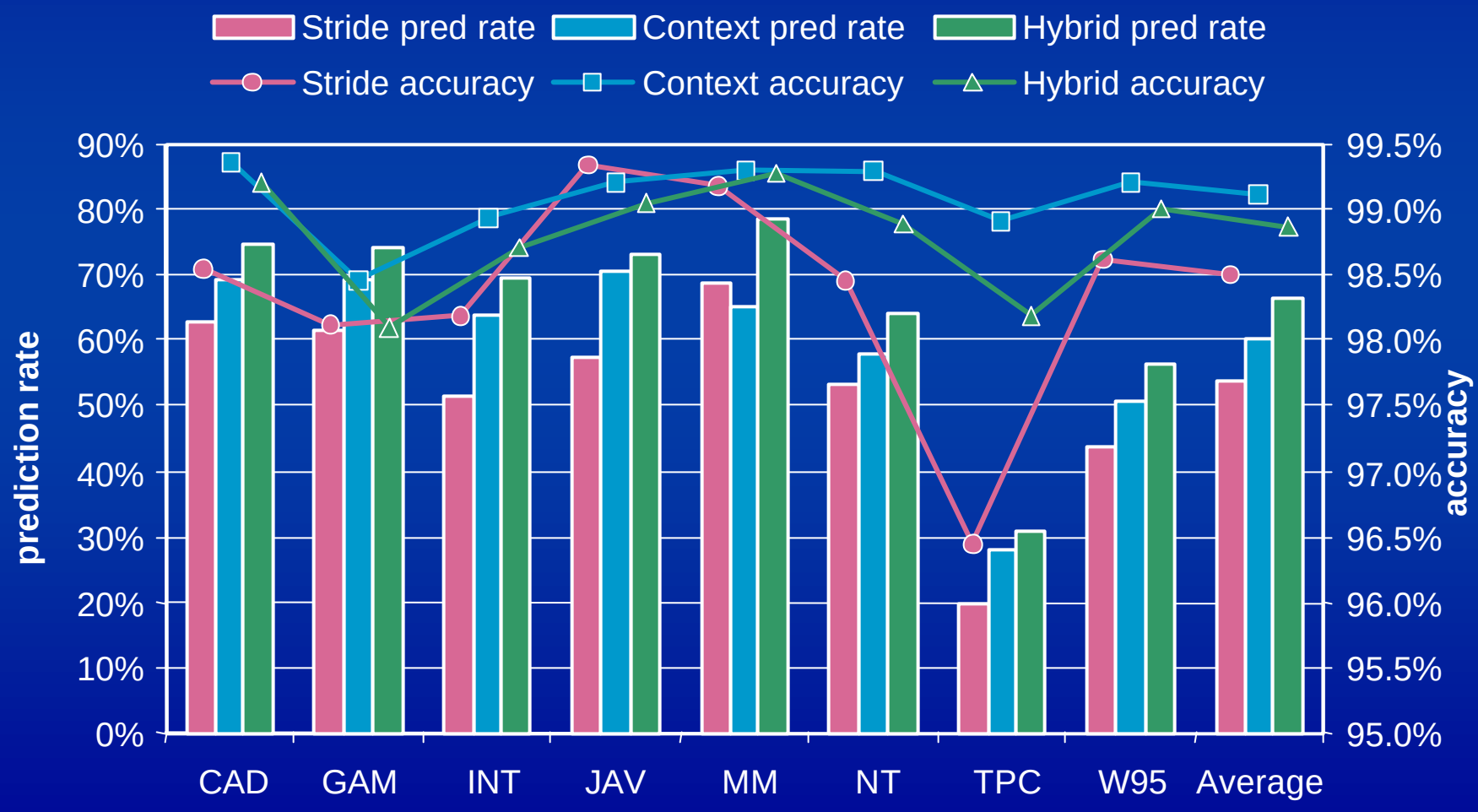


Simulation: Processor Configuration

- 45 traces from 8 different benchmarks
- 8-wide out-of-order, 128 instruction ROB
- 6 INT Execution Units, 4 Data Cache ports
- Instruction latencies common to Intel processors
- 32KB L1 Dcache, 1MB unified L2
- Aggressive Instruction Fetch
- Hybrid Branch Predictor
- Dynamic memory disambiguation

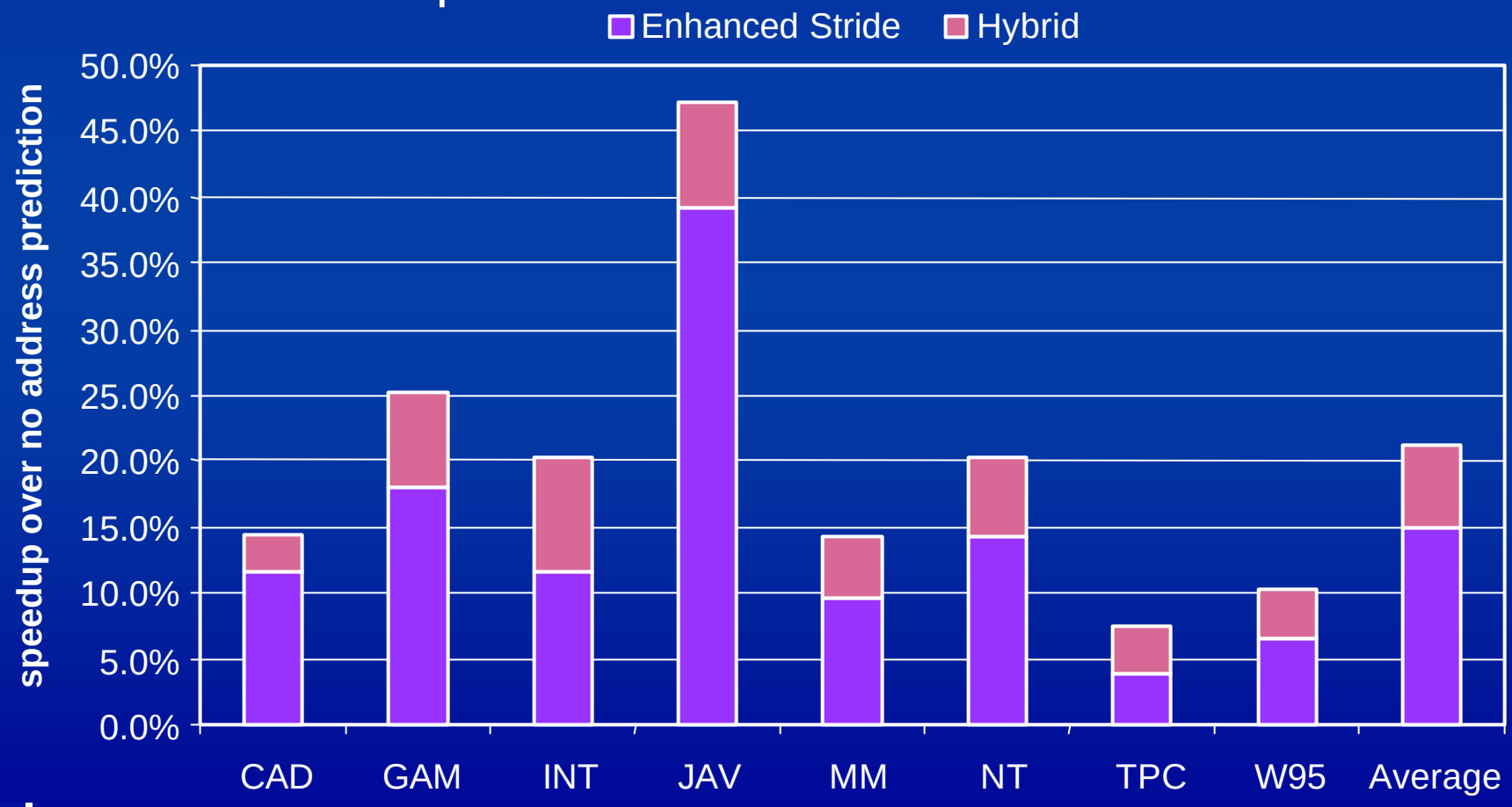
Hybrid Predictor Performance

- Predicts 67% of all loads with 99% accuracy
  - Immediate update



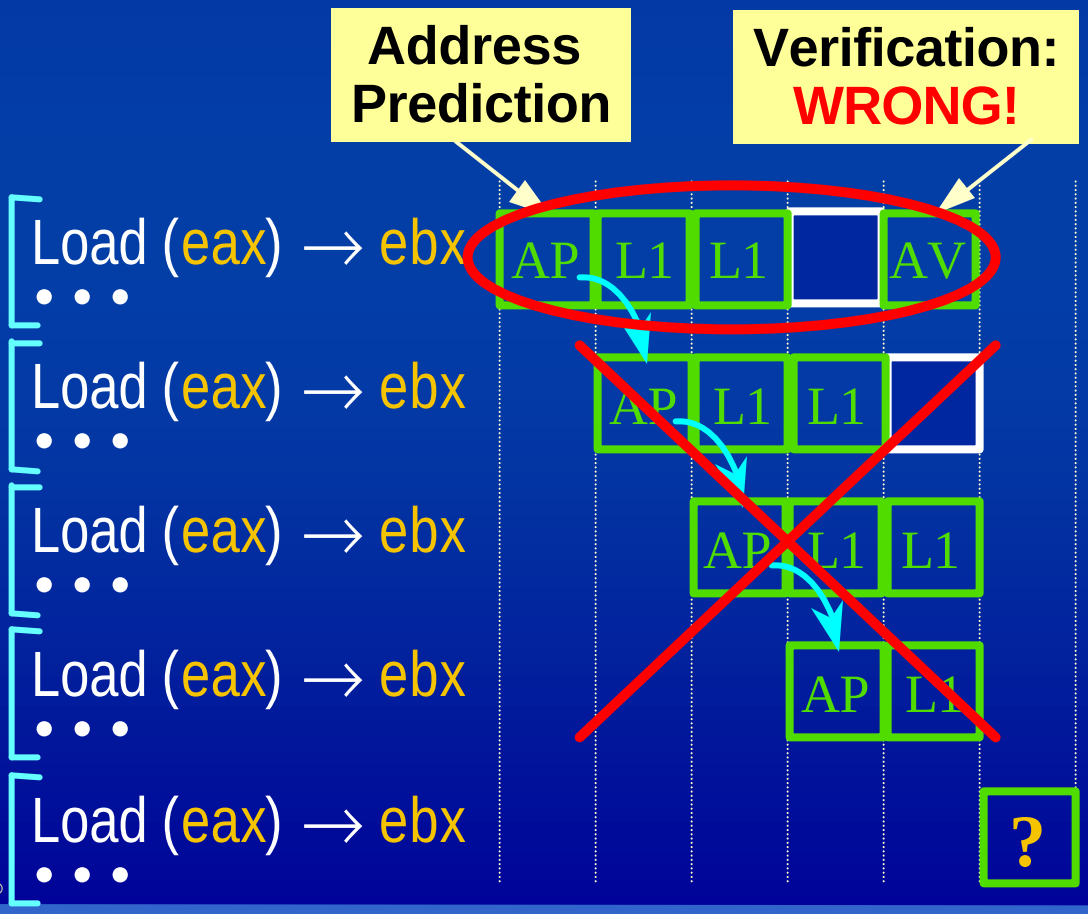
Speedup

- Stride - 14.7%, Hybrid - 21%
  - Aggressive processor configuration
  - Immediate update



Pipeline Issues

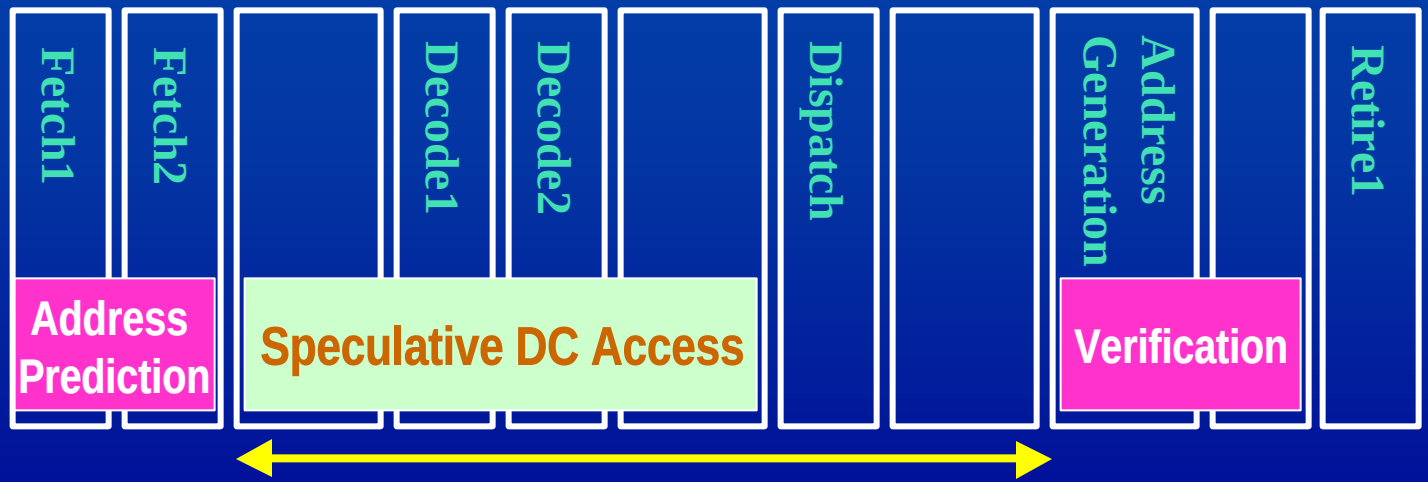
- Multiple pending unresolved predictions
  - Successive predictions are correlated
    - Stride: Last Address
    - Context-based: History
  - A misprediction is propagated to all pending predictions



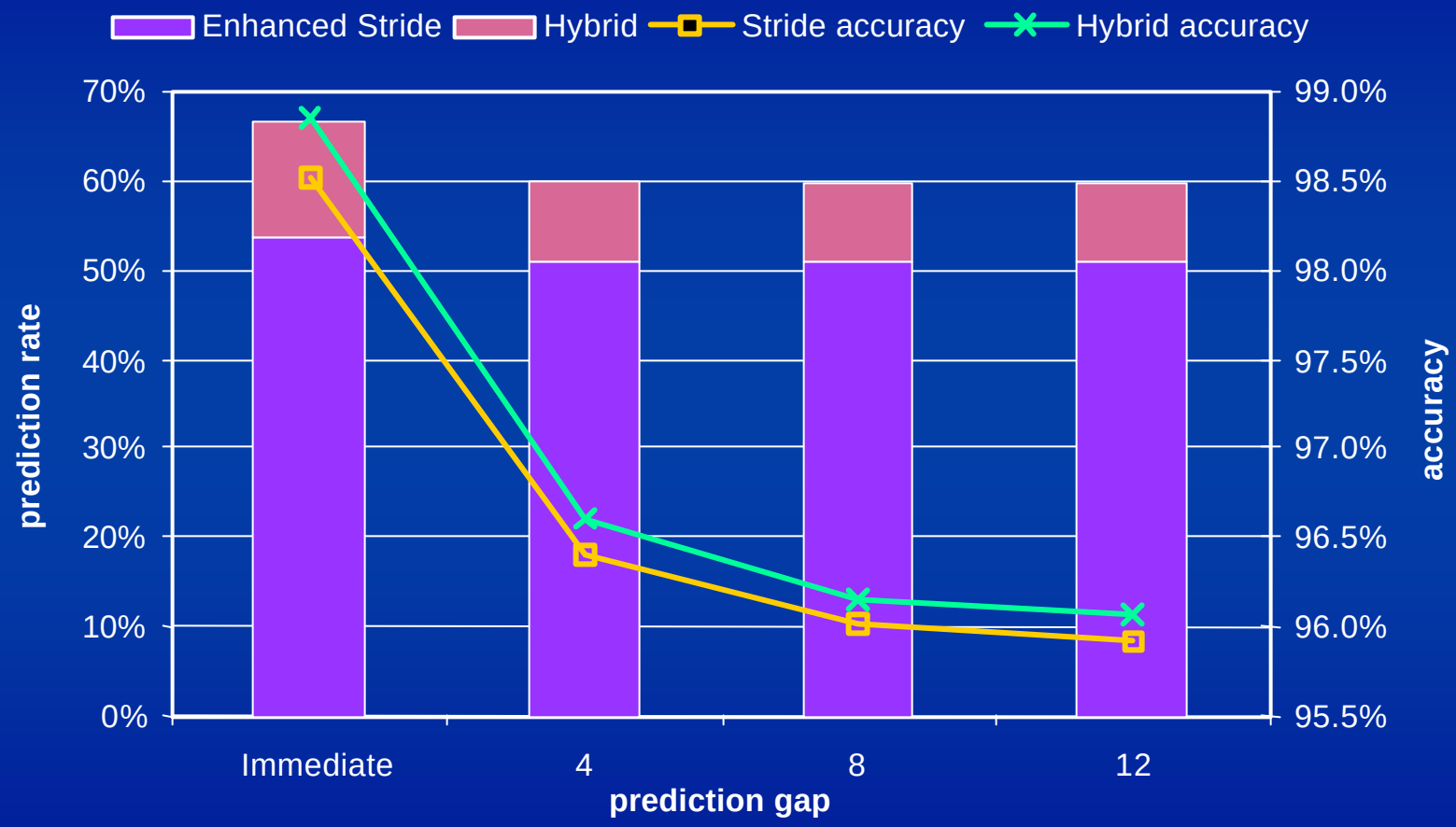
- Catch-up
  - Stride: extrapolate
  - Context-based: stop predicting
- Disruptive events enable prediction restart

Pipeline Issues

- Pending predictions of the same static load
  - Must update the structures at prediction time
  - Update may turn out to be wrong at verification time
- Updates on the wrong control path
- Predictors use Address Reorder Buffer to maintain speculative state

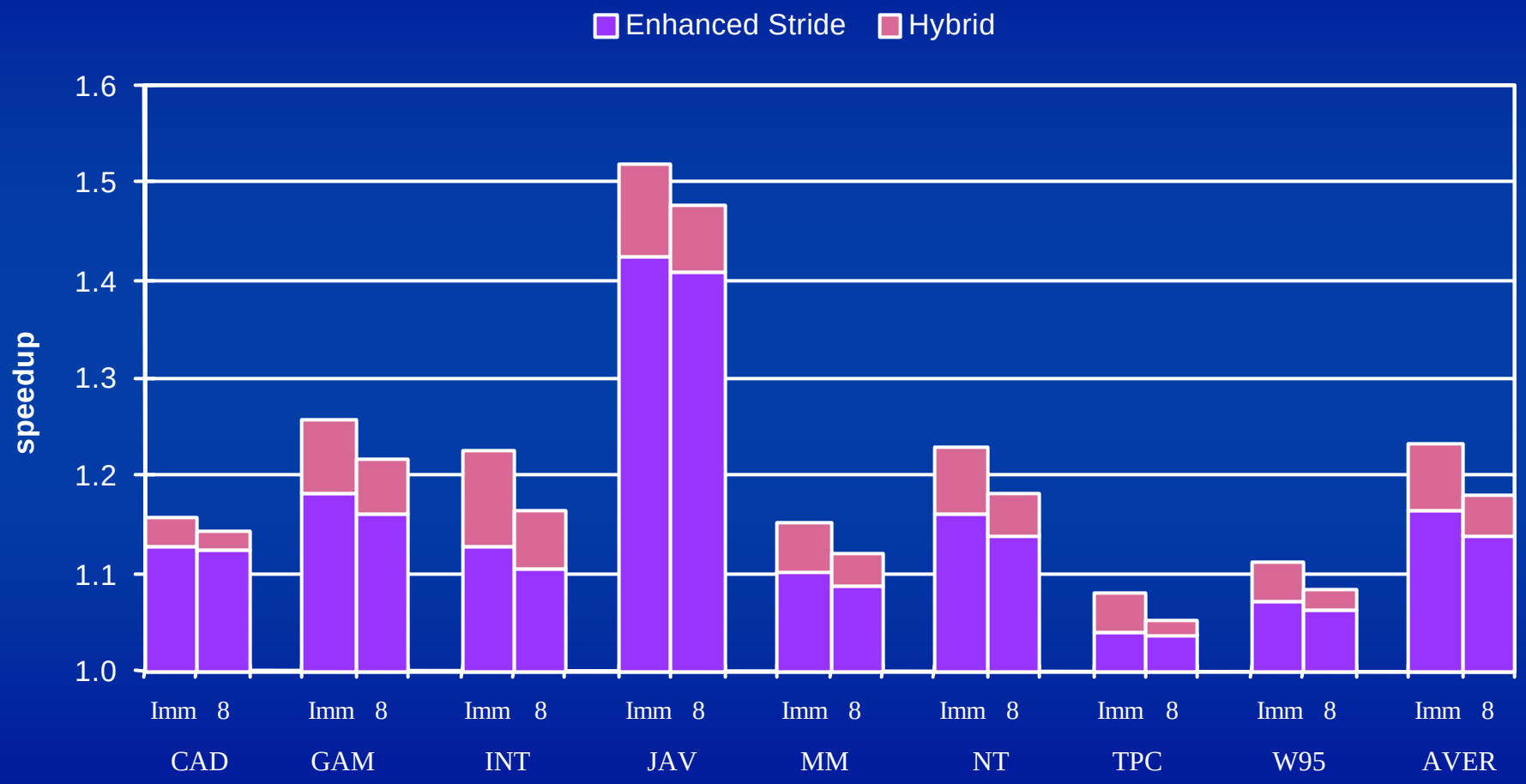


Influence of Pipeline Length on the Prediction Rate



- Increasing the pipeline length impacts Accuracy more than the Prediction Rate

Influence of Pipeline Length on Overall Performance



- Prediction Gap of 8 cycles vs. Immediate Update

Summary

- Introduced an efficient context-based address predictor
  - Global correlation
  - Advanced confidence mechanisms
  - Pollution-eliminating structures
- Constructed a hybrid predictor to improve cost & performance
- Explored mechanisms required for Address Prediction in a pipelined environment
  - *Correct* prediction rate of 57% with prediction gap of 8 cycles vs. 66% with immediate update
- Future work
  - Prediction schemes for the remaining loads
  - Load classification